컴포넌트 기반 회계처리 응용 시스템 개발에 관한 연구

이정직*, 류상훈**
고려대학교 소프트웨어공학과* , 고려대학교 컴퓨터학과**
jjlee@korea.ac.kr*, shryu@swsys2.korea.ac.kr**

A Study on the Component-Based Development For Accounting Process Application System

Jeong-Jig Lee* , Sang-Hoon Ryu**
Dept. of Software Engineering, Korea University*
Dept. of Computer Science & Engineering, Korea University**

요 약

CBD(Component Based Development)는 장기간에 걸쳐 발전된 소프트웨어 개발의 한 형태이며, 이미 표준화되거나 공인되지 않았지만 실제로 모든 기업들은 컴포넌트를 나름대로 제작하거나 이미 잘 개발되어진 컴포넌트를 재사용하여 자체 프로젝트에 재활용하므로써 최소의 개발비로 표준화를 추구하는 한편 최대의 기능과 신속한 개발로 소프트웨어의 품질과 생산성 향상을 도모하고 있다.

본 논문에서는 레거시 시스템를 활용하여 컴포넌트로 되어 있지 않은 프로그램을 컴포넌트화 함은 물론 기존 애플리케이션에 존재하는 컴포넌트를 래핑하여 재사용 가능한 서비스를 생성, 재공하기 위한 CBD 기반의 회계처리 응용 시스템을 설계 및 구현한다. 제안된 회계처리 응용 시스템 APAS(Accounting Process Application System) 모델은 시스템 구축 개발 기간의 단축과 관리 및 유지보수를 쉽게 하여 업무의 효율성을 높이고, 레거시 시스템을 재활용하여 신규 개발시 우려되는 위험도를 낮추어 소프트웨어 생산성 제고 효과도 얻을 수 있다.

1. 서 론

IT(Information Technology)환경이 복잡하고 빠르게 변화함에 따라서 구축될 시스템도 이러한 환경에 대처하기 위해서 제한된 시간에 최소한의 비용으로 원하는 시스템을 구축하는 것이 필수적이다. 또한 사용자의 요구사항이 새롭게 변경될 때마다 소프트웨어에 그 변경이 반영되어야 하며 이러한 변경이 소프트웨어에 반영하는 데는 많은 시간과 비용이 소요된다. 따라서 사용자의 요구사항 변경에 따른 반영, 빠른 시스템 구축, 유지 보수 단계의 효율적인 시스템 관리, 소프트웨어의 수정 용이성, 저렴한 비용 등으로 인해 컴포넌트 기반 시스템 구축이 필수적으로 요구된다. 또한 기업에서는 성공적으로 프로젝트를 진행하기 위해 기존 소프트웨어의 재사용과 유지 보수의 필요성을 느끼게 되었으며, 이를 위해서 전반적인 소프트웨어의 설계나 원시 코드의 생성 등에서 유지 보수를 위한 소프트웨어의 이해는 물론, 컴포넌트로 되어 있지 않은 프로그램을 컴포넌트화 함으로써 재사용성을 높여주는 일 등이 필요하다.[1]

본 논문에서는 레거시 시스템를 활용하여 컴포넌트로 되어 있지 않은 프로그램을 컴포넌트화 함은 물론 기존 애플리케이션에 존재하는 컴포넌트를 래핑하여 재사용 가능한 서비스를 생성, 재공하기 위한 CBD 기반의 회계처리 응용 시스템을 설계 및 구현한다. 이 시스템의 효율성을 검증하기 위하여 회계처리 응용 시스템 APAS(Accounting Process Application System) 모델을 제시한다. 이 모델의 중요 부분들을 플랫폼과 시스템으로 나누어 컴포넌트로 구현하고, 이를 이용하여 관리자용 서버 시스템을 구축하고, 이 시스템을 사용하는 클라이언트 프로그램들을 구현한다.

2. 관련 연구

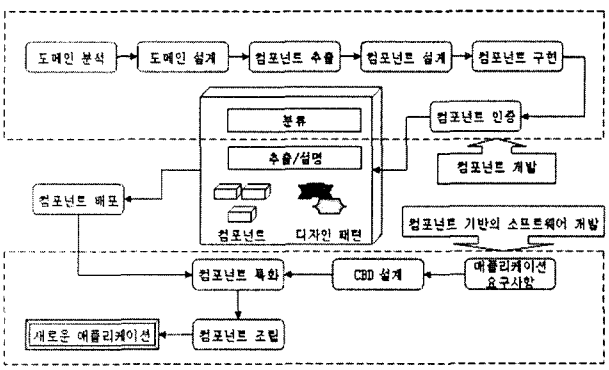
2.1 레거시 시스템을 이용한 역공학 방법

레거시 시스템(Legacy system)은 조직(Organization)의 데이터 처리를 위해 오랜 기간동안 개발되고 만들어진 코드와 데이터, 기술들을 말한다. 이것은 기본적인 프로세서, 중요한 데이터, 백본 네트워크, 관련 기술, 개발에 사용된 과거의 언어들을 포함한다. 이러한 레거시 시스템에서 나타나는 몇가지 문제점들을 보면, 첫째, 레거시 시스템은 개발 문서가 없는 경우이거나 형식에 그친 문서들뿐이다. 이는 레거시 시스템에 대해서 알려진 지식이 불충분하고, 그 결과 시스템에 대한 이해가 부족하게 되어 그 시스템의 성장과 발전이 어렵게 되는 원인이 된다. 둘째, 유지보수가 어렵다는 것이다. 특히, 기능(function)을 보다 관리가 용이한 구성 요소(component or object)들로 분할할 수 없다. 셋째, 대부분의 기업에서 사용하고 있는 레거시 시스템들은 80년대에 개발된 기술을 사용하고 있는 경우가 대부분이다. 이는 호스트 중심의 폐쇄형 시스템(Closed System)이므로 시스템 통합이 어렵다. 이와 같이 레거시 시스템은 일체적이고, 수직적으로 통합된 애플리케이션들이며, 사용자 독자적인 방법을 적용한 폐쇄형 시스템이다. 또한 소프트웨어의 구조가 복잡하여 개발자나 시스템 관리자가 유지 보수를 하는데 있어 어려움이 있으며, 재사용과 확장성의 기능이 부족하고 시스템 개발 및 배포시 많은 시간이 소모된다. 이는 유지보수 및 시스템 진화에 많은 비용이 드는 결과를 초래한다.[2] 이러한 레거시 시스템으로부터 의미 있는 정보를 추출해내는 방법중 역공학(Reverse Engineering)을 이용한 방법이 대표적인 예라 할 수 있다. 역공학 방법의 핵심은 레거시 시스템으로부터 단지 의미 있는 정보만을 추출하는 것이 아니라, 추출된 의미 있는 정보를 보다 상위수준의 추상화로 유도하는 것이다.[3] 레거시 시스템은 Hyper 문서, 오브젝트 모델, 타

입, 시각적 요소, 컴포넌트 추출 및 변환하기 위한 전문적인 정보를 바탕으로 새로운 시스템으로 재개발 된다.

2.2 컴포넌트 기반 개발(CBD : Component Based Development)

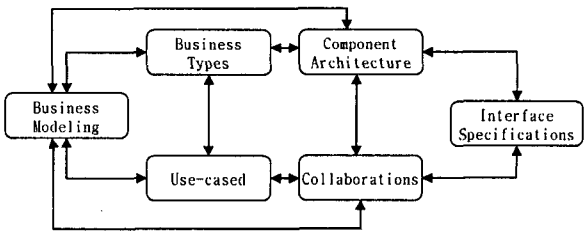
컴포넌트 기반 개발은 독립적인 기능을 담당하는 다양한 컴포넌트 소프트웨어의 집합으로부터 해당 업무의 수행에 필요한 기능을 담당하는 하나 이상의 컴포넌트를 결합함으로써 해당업무를 위한 소프트웨어를 개발하는 기술을 말한다.[4] 컴포넌트 기반 개발 기술은 과거 구조적 방법이나 객체 지향 기술이 제대로 해결하지 못한 소프트웨어 생산성, 소프트웨어 재사용성, 시스템 유지보수성을 향상시킬 수 있는 대안으로 주목받고 있다.[5] CBD에는 두가지 형태의 개발 방법이 있다. 그 중 하나는 컴포넌트 자체를 개발하는 CD(Component Development)이고, 다른 하나는 이미 구축되어 있는 컴포넌트를 사용하여 소프트웨어 시스템을 개발하는 CBSD(Component Based Software Development)이다. 컴포넌트 개발(CD)은 완전한 소프트웨어 시스템을 만드는 것이 아니라 다른 소프트웨어 시스템에 포함될 부품을 만드는 것이다. 여기에서 개발되는 컴포넌트는 여러 소프트웨어 시스템 개발에 재사용될 목적으로 만들어야 하기 때문에 해당 업무 도메인에 대한 분석이 매우 중요하다. 다양한 소프트웨어 시스템에 공통적으로 필요한 기능을 가진 컴포넌트를 개발해야 재사용성이 높아지게 되어 컴포넌트 기반 개발 방법이 효과를 발휘할 수 있다. 컴포넌트 기반 소프트웨어 개발(CBSD)은 이미 개발된 컴포넌트들을 조립하여 비즈니스의 요구사항을 충족시키는 소프트웨어 시스템을 개발하는 것을 말한다. 이 개발 과정에서 중요한 것은 소프트웨어 시스템을 처음부터 개발하는 것이 아니라, 이미 잘 개발되어 있는 컴포넌트 즉, 부품들을 그대로, 또는 특화(Customization)시킨후 컴포넌트를 조립하여 소프트웨어 시스템을 개발하자는 것이다. 이것이 소프트웨어 시스템의 개발 시간을 단축시킬 뿐만 아니라, 소프트웨어 시스템 개발에 필요한 인력이나 비용의 절감 효과를 가져올 수 있다. (그림 1)은 컴포넌트 개발과 컴포넌트 기반의 소프트웨어 개발 과정을 나타낸 것이다.[6]



(그림 1) 컴포넌트 개발과 컴포넌트 기반 소프트웨어 개발 과정

2.3 컴포넌트 기반 개발 모델링

CBD 모델링 기술은 객체지향의 UML(Unified Modeling Language) 분석 및 설계 모델링에 대한 표기법과 의미를 이용한다. 컴포넌트 기반 개발 모델링에는 비즈니스 모델링, 비즈니스 타입 모델링, 유즈케이스 모델링, 컴포넌트 아키텍처 모델링, 상호협력 모델링 등이 있다. (그림 2)는 컴포넌트 기반 개발 모델링 맵이다.[4-7]

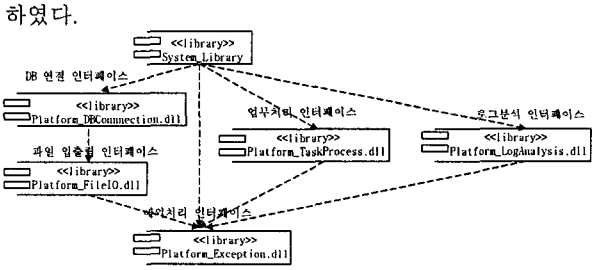


(그림 2) 컴포넌트 기반 개발 모델링 맵

3. 설계 및 구현

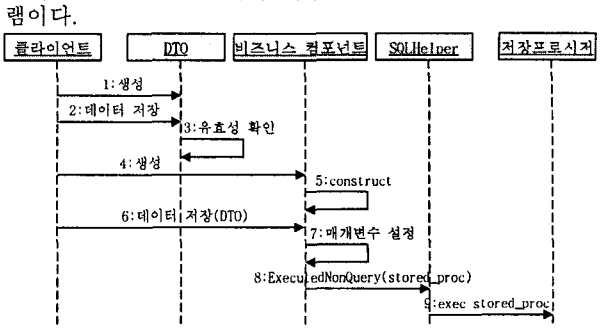
3.1 APAS 설계

APAS의 분석 및 설계 방법으로 UML을 이용한 컴포넌트 다이어그램과 시퀀스 다이어그램을 통해서 설명한다. APAS 플랫폼은 시스템 구축에 필수적인 요소를 중심으로 하여 컴포넌트화 하였다. (그림 3)은 APAS 플랫폼 컴포넌트 다이어그램이다. 플랫폼을 데이터베이스 연결 인터페이스, 업무처리 인터페이스, 로그분석 인터페이스, 예외처리 인터페이스, 파일 입출력 인터페이스 등으로 구성하여 요청될 서비스의 기능을 중심으로 컴포넌화 하였다.



(그림 3) ASAP 플랫폼 컴포넌트 다이어그램

(그림 4)는 APAS 데이터 저장프로시저 시퀀스 다이어그램이다.



(그림 4) 데이터 저장프로시저 시퀀스 다이어그램

클라이언트는 커스텀 DTO 클래스의 인스턴스를 생성하고 데이터를 저장한 후 데이터 유효성을 확인한다. 클

라이언트가 비즈니스 컴포넌트의 인스턴스를 생성하고, 데이터베이스 연결 문자열 정보를 읽어온다. 클라이언트가 커스텀 DTO 클래스의 인스턴스를 매개변수로 비즈니스 컴포넌트의 데이터 저장 메서드를 호출하면, 저장 프로시저의 매개변수를 설정하고, SqlHelper 클래스의 ExecuteNoQuery메서드를 호출하여 저장프로시저를 실행하여 데이터를 저장하는 것을 보여주고 있다.

3.2 APAS 구현

본 논문에서는 레거시 시스템를 활용하여 컴포넌트로 되어 있지 않은 프로그램을 컴포넌트화 하고 기존 애플리케이션에 존재하는 컴포넌트를 래핑하는 방식으로 구현하였다. 닷넷 기반으로 구현된 각각의 컴포넌트들은 실제로 구현될 서비스 기술에 따라 서비스를 제공하는 역할을 담당할 수 있도록 쉽게 변경 가능하며 모듈 변경 시, 해당 모듈만 수정함으로써 프로그램 코드의 유지 보수성을 높여 응용 시스템 개발의 생산성을 향상시킬 수 있다.

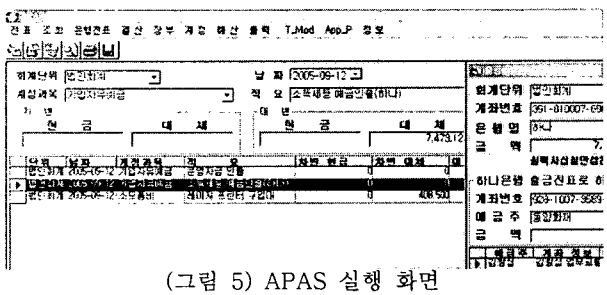
3.2.1 구현 알고리즘

APAS 의 핵심 알고리즘은 APAS_SERVICES 와 APAS_SYSTEM 으로 구성된다. 다음은 서비스 제공자 측이 서비스를 요청자에게 제공하는 로직을 처리하는 APAS_SYSTEM 알고리즘의 일부분이다.

```
private bool performDataUpdate(string connection_
String,string creditAccountNumber,decimal amount)
{
 using ( SqlConnection conn = new SqlConnection(
            connectionString))
 {
  conn.Open();
  // Establish command parameters
  // @AccountNo (To Account)
  SqlParameter paramToAcc = new SqlParameter(
              "@AccountNo",SqlDbType.Char, 30);
  paramToAcc.Value = creditAccountNumber;
  // 생략...
```

3.2.2 시스템 구현

(그림 5)는 구축된 시스템의 실행 화면을 보여주고 있다.



(그림 5) APAS 실행 화면

3.2.3 시스템 환경

제안 시스템은 [표 1]의 기술을 이용하여 서비스를 제공할 수 있도록 구현하였다.

환 경	구현 기술
Server OS	Windows 2003 Server
Client OS	Windows 계열
Program Language	VS.Net 2003
Database Server	Ms-SQL 2000

[표 1] 시스템 환경

4. 결론

본 논문에서는 컴포넌트 기반의 회계처리 응용 시스템인 APAS를 제안하여 설계 및 구현하였다. 제안된 시스템은 클라이언트에서 시스템에 대한 전체적인 이해없이 제공되는 인터페이스에 대한 이해만으로 프로그램 구현이 가능하였다. 또한, 제안된 모델을 이용하여 시스템을 구축함으로써 개발 기간의 단축을 가져왔고, 컴포넌트화 함으로써 관리 및 유지보수가 쉬워 업무의 효율성을 높이게 되었다. 레거시 시스템을 재활용하여 신규 개발시 우려되는 위험도를 낮추었고, 소프트웨어 생산성 제고 효과도 얻을 수 있었다. 뿐만 아니라 표준화를 선행하여 시스템을 구축함으로써 시스템 통합도 손쉽게 가능하게 하였다.

향후 연구과제로는 서비스를 확장시 성능을 최대로 높이는 연구가 이루어져야 한다. 이를 위해서 자주 요청되는 서비스는 캐시에 저장해두어 요청시 캐싱하여 리턴하므로써 성능을 향상시킬 수 있게 하고, 특정한 입력 매개변수에 따라 결과가 달라지거나 입력 매개변수 값의 범위가 결정되지 않은 서비스 메서드는 캐싱이 적합하지 않으므로 비동기 메서드 호출(asychronous method call)방식으로 시스템의 성능을 높일 수 있는 연구가 필요하다.

참 고 문 헌

[1] Jeffrey Voas, "Maintaining component-based system", IEEE Software, July/August 1998.

[2] A. J. O'Callaghan, "MIGRATING LARGE-SCALE LEGACY SYSTEMS TO COMPONENT-BASED AND OBJECT TECHNOLOGY : The Evolution of a Pattern Language" , De Montfort University, Leicester, United Kingdom, CAIS, Vol 2, Article 3, July 1999

[3] E. J. Chikofsky and J. H. CrossⅡ, "Reverse Engineering and Design Recovery : a Taxonomy", IEEE Software, pp. 13~17, 1990

[4] 차정은, "컴포넌트 기반 개발 프로세스 지원을 위한 컴포넌트 저장소의 설계 및 구현", 대구카톨릭대학교 전산통계학 전공 박사학위논문, Feb. 2001

[5] 조완수, "UML 객체 지향 분석·설계", 홍릉과학출판사, Apr. 2000

[6] 전병선, ".NET Enterprise System 객체지향 CBD 개발 방법론", 영진닷컴, Jun. 2004

[7] Alan Brown, "Using service-oriented architecture and component-based development to build web service applications", Rational Software white pater from IBM, Apr. 2002